Ada 2005

Richard L. Conn
Retired (formerly of Microsoft and Lockheed Martin)

Since its formulation in the 1970s, Ada has had a significant impact on the future of government and commercial safety-critical applications as well as the development of other computer programming languages and environments. This article discusses the creation and new key features of Ada 2005, and compares it to other computer programming languages.

The needs of the Department of Defense (DoD) and the capabilities offered by the Ada programming language to support the development of mission-critical, software-intensive systems have matched, by design, for more than two decades. Control of the definition of Ada has passed from the DoD to the American National Standards Institute[1] and then to the International Organization for Standardization (ISO)[2]. Likewise, support for the Ada community has passed from the DoD to a community of users and vendors, thereby eliminating the cost of support for the DoD. The needs of the DoD have changed throughout the years, and the stewards of Ada within ISO have seen to it that Ada changes as well. Operating in an open forum with extensive opportunity for user feedback, the stewards of Ada have caused it to evolve to continue to meet those needs as well as the needs of an international community of users.

Many languages, such as Sun's Java[3], Microsoft's C#[4], and Visual Basic[5], are owned and controlled by companies. Languages such as Ada[6], C[7], and C++[8] are instead defined by ISO with no direct control or enforcement mechanism other than the free market. Sometimes both corporate and ISO have control and ownership in some form, as is the case with Microsoft's C#[9] and Common Language Infrastructure[10]. All languages are impacted by user feedback because of their market ties, but the question of ownership and control can make a decisive difference in language selection within certain application domains. Developers in domains driven by short-term market needs and fluctuations are often happy to follow the latest trends and let others worry about infrastructure support, including computer language support. Developers in domains driven by long-term (sometimes covering many decades) product life cycles may need languages and environments that are more stable in the long run. Its long-term stability and ISO-controlled evolution, as well as its support for safety-critical, high-integrity, large-scale systems development can make Ada attractive to users developing systems with a long life cycle such as the DoD.

Today, Ada is used in many domains in many ways. A theme in several of these domains is that the systems using Ada have long life cycles, are safety-critical, are mission-critical, are large (often containing several million lines of code), and they require high levels of reliability. So it is not surprising that we are also finding Ada in use in several future systems. Ada continues to be with us (no, Ada is *not* dead as some people have come to believe), and it continues to evolve in the form of Ada 2005. ISO's definition for Ada 2005 was *put to bed* in November 2005 and should be officially approved through an international ballot in November 2006. This article presents an overview of Ada 2005 and presents information on how it relates to previous versions of Ada and other computer programming languages.

> *"... the systems using Ada have long life cycles, are safety-critical, are mission-critical, are large (often containing several million lines of code), and they require high levels of reliability."*

Stewards of Ada

Ada has been developed in an international open forum sponsored by ISO and the International Electrotechnical Commission (IEC), specifically ISO's Joint Technical Committee (JTC) 1. JTC1 is assigned to deal with all standardization activities in the domain of information technology. As of March 2006, 28 countries participate in JTC1, with another 42 countries registered as observers; JTC1 is responsible for 538 ISO standards. JTC1 is divided into subcommittees (SCs). SC22 deals with *programming languages, their environments, and system software interfaces*; SC22 is divided into working groups (WGs). WG9 is responsible for the *development of ISO standards for the programming language Ada*. The people participating in ISO/IEC JTC1/SC22/WG9, which include users, vendors, and language lawyers, are the stewards of Ada; James W. Moore of the MITRE Corporation is the convener of WG9. People from the following countries are most actively involved in WG9: Canada, France, Germany, Japan, the Netherlands, Russia, Spain, Switzerland, the United Kingdom, and the United States. For more information on ISO and Ada 2005[11], visit <www.iso.org/iso/en/ISOOnline.frontpage>.

Overall Goals for Ada 2005

There are two overall goals for Ada 2005:

- Enhance Ada's position as a safe, high-performance, flexible, portable, interoperable, concurrent, real-time, and object-oriented programming language.
- Further integrate and enhance the object-oriented capabilities of Ada.

As always, upward compatibility with previous versions of Ada is a prime concern. Note that Ada 2005 will be published as an *amendment* to Ada 95, not a new language.

Over the decades, the developers of programming languages have been learning from each other. Ada has influenced the development of Java, C++, Visual Basic, and even the Microsoft .NET Framework. Likewise, Ada has been influenced by more than 30 other languages, including Java, C, and C++. The stewards of Ada designed Ada 2005 to offer the following:

- At least the safety and portability of

Java.
- At least the efficiency and flexibility of C and C++.
- An open, international standard for real-time and high-integrity system development.

Ada can also take advantage of the enormous libraries of reusable components created by the developers of other languages and computing environments. Dr. Martin Carlisle[12] of the U.S. Air Force Academy, for example, has created the A# compiler, which can readily make use of the Microsoft .NET Framework class libraries and create code in the Microsoft Intermediate Language for compilation by the just-in-time compiler in the Microsoft .NET Framework.

Some Key New Features of Ada 2005

The key new features of Ada 2005 reflect both a *catching up* to update Ada with ideas that have become popular with other programming languages, and a *leaping ahead* to enhance some features of Ada that are more unique to Ada itself.

Safety First

Some people consider Ada to be the premier language for safety-critical software, so the Ada 2005 amendments have been carefully designed to not open any safety holes. In addition, some amendments provide even more safety, in some cases putting even more load on the Ada compiler for catching defects at compile time.

Object.Operation Notation

The familiar and popular *object.operation* notation employed in the most popular languages (such as C++, C#, Java, and Visual Basic) is now available in Ada 2005. Programmers of Ada 2005 may use either the *operation* (object, parameters) mechanism required by Ada 95 or the *object.operation* (parameters).

Extensions to the Open, Predefined Ada 95 Library

Ada 2005 adds the following new standard packages that provide features already found in the implementations and libraries of other languages:
- Environment variables.
- Time access and manipulation, including calendar arithmetic and time zones (including predefined types like DAY_NAME and YEAR_NUMBER).
- File and directory manipulation.
- Containers and sorting (including a predefined generic array sort).
- More string functions and wider characters (type WIDE_WIDE_CHARACTER).
- Linear algebra.

New Features for Real-Time and High-Integrity Systems

As a language to support safety-critical, high-integrity systems, Ada 2005 adds several new standard features that have been informally employed by active Ada software developers in the past, reflecting user interests and needs:
- Earliest deadline first, real-time scheduling.
- Round-robin, real-time scheduling.
- The Ravenscar high-integrity, run-time profile[13].

The Ravenscar profile promotes simple and effective language-level concur-

Safety-Critical Issues and Legacy

A recent report puts the cost of downtime at more than $6 million per hour for financial markets[1]. The report lists this as the extreme value of downtime (with shipping downtime being the low end, at $28,000 per hour). However, in the Department of Defense (DoD), the cost of an hour of downtime could easily be measured not in dollars, but instead measured in human lives. As a result, extreme efforts have to be taken to ensure that our safety-critical software does not suffer downtime. To this end, safety-critical languages have become important, especially in DoD-related software[2]. While this comes as no great shock to programmers that C and C++ are not considered safety-critical languages, Java has matured into a real-time language that is robust enough for real-time safety-critical applications[3]. It is worth noting that many have said that Java closely resembles C++ syntax with Ada semantics[4].

Ada is very much a viable language choice for safety-critical embedded and real-time systems. Some examples follow:
- The C-130J (Hercules) aircraft, manufactured by Lockheed Martin. The C-130J has been adapted for roles such as airlifters, hurricane hunters, tankers, and electronic surveillance.
- The F/A-22 (Raptor) advanced tactical fighter, manufactured by Lockheed Martin in collaboration with Boeing. The stealthy F/A-22, which just recently achieved operational deployment, is the most advanced fighter in the world, used exclusively by the United States.
- The F-16 (Fighting Falcon) fighter, manufactured by Lockheed Martin. The F-16 (the world's first fourth-generation fighter) is the most widely used fighter in the world, with more than 4,000 F-16's deployed by 24 countries in 110 versions.
- The F-35 (Joint Strike) fighter, manufactured by Lockheed Martin. The stealthy F-35 is a multi-role fighter for the next generation, designed for use by the Air Force, Navy, and Marines as well as many allies of the United States.
- The A380 family of aircraft, manufactured by Airbus. The environmentally friendly, fuel-efficient A380 passenger jet airliner is the largest passenger jet in existence, seating as many as 555 people in comfort.
- Many other commercial aircraft. These include the Boeing 777 plus assorted Airbus and Embraer aircraft, as well as future aircraft such as the Boeing 7E7 prototype.
- The National Ignition Facility (NIF) at Lawrence Livermore National Lab. The NIF houses the largest LASER facility in the world, used in experiments in high-energy density and fusion technologies with direct applications to nuclear stockpile stewardship, energy research, science, and astrophysics.
- Air Traffic Control system of the United States. Key elements of the Air Traffic Control system are used in the United States, 60 other countries, the subways of New York City and Paris, unmanned vehicles (ground, aerial, and submersible), and more.

These systems are not only safety-critical systems, but they have a very long lifetime (in terms of decades). As such, legacy issues are raised. These systems must evolve as their requirements change, as their missions change, and as lessons learned on topics develop, helping them become more secure and reliable. It is vital that the software they use also evolve with them, and Ada is positioned to do just that.

Notes

1. See <www.cnsoftware.org/nss2report/Chen-NSS2v.3.pdf>.
2. See "Correctness by Construction: A Manifesto for High-Integrity Software" by Martin Croxford and Roderick Chapman, CROSSTALK, Dec. 2005 at <www stsc.hill.af.mil/crosstalk/2005/12/0512CroxfordChapman.html>.
3. See <www.rtj.org/rtsj-V1.0.pdf>.
4. See "Software Standards: Their Evolution and Current State" by Reed Sorensen, CROSSTALK, Dec. 1999 at <www.stsc.hill.af.mil/crosstalk/1999/12/ sorensen.asp>.

rency, which is essential for safety-critical applications. It is a subset of the Ada 95 tasking model, which contains restrictions to meet real-time community requirements for the following:

- Determinism.
- Schedulability analysis.
- Memory-boundedness.
- Execution efficiency.
- A smaller memory footprint.
- The need to satisfy certification requirements, such as Federal Aviation Administration Aircraft-type certification.

The Ravenscar profile includes, but is not limited to, the following set of restrictions and features:

- No task entries are allowed.
- A maximum of one protected entry is allowed.
- No abort statements are allowed.
- No asynchronous control is allowed.
- No dynamic priorities are allowed.
- No implicit heap allocations are allowed.
- No task allocators are allowed.
- No task hierarchy is allowed.
- The maximum length of an entry queue is one.
- Protected types are not allowed.
- Relative delays are not allowed.
- Requeue and select statements are not allowed.
- Task termination is not allowed.
- User-defined timers and local timing events are not allowed, but execution timers (to help catch task overruns) are allowed (and predefined in the Ada 2005 library).

Popular Interface Approaches

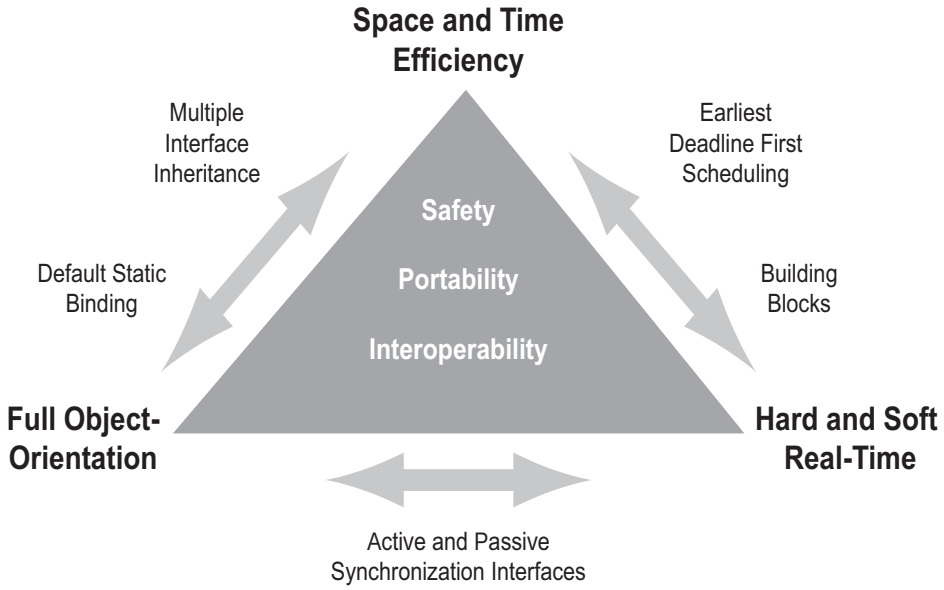
Ada 2005 now supports the notions of interface used in languages such as Java and C#, and architectures such as CORBA. To address a common user need, Ada 2005 adds a new pragma called *Unchecked_Union* for interoperating with C and C++ libraries. Interface types have been added, and Ada 2005 *leaps ahead* with the notion of both active and passive synchronized interface types that efficiently integrate object-oriented programming concepts with real-time programming concepts.

Enhanced Encapsulation

Ada 2005 fully supports both module and object encapsulation. With module encapsulation (as already supported by Java), no synchronization is implied, and access within the module is restricted to private components of a type to the module in which the type is declared (that is, you cannot refer to the internals of a module outside of that module). Private components of multiple objects may be referenced simultaneously. With object encapsulation (as already supported by Eiffel), access to an individual object is synchronized. Only operations inside a protected or task type can manipulate components of a locked object.

"Ada has influenced the development of Java, C++, Visual Basic, and even the Microsoft .NET Framework. Likewise, Ada has been influenced by more than 30 other languages, including Java, C, and C++."

Figure 1: *Ada 2005: Putting It All Together*



Access Types Enhanced

Ada 95 has been considered too rigid by some in its definition of access types. In many cases, a significant number of explicit conversions are required to access anonymous objects and parameters. Ada 2005 adds *anonymous access types* to remove the need for so many conversions.

Dependency Issues Resolved

Addressing a fairly common user need, Ada 2005 adds support for cyclic dependence between types in different packages. *Limited with* clauses allow a limited view of a package, thereby permitting types to be defined across package boundaries.

Figure 1 shows the three key areas addressed by Ada 2005 (full object-orientation, space and time efficiency, and hard and soft real-time requirements), and the key new features related to them such as *earliest deadline first* scheduling.

Ada Resources

A number of compilers, development platforms, tools, training and education aids, reusable software components, and other resources are available through the global community of Ada users and vendors. Migration in several forms is now taking place to support development using Ada 2005, and the single best starting point for engaging with the Ada community is the SIGAda Web site at <www.sigada.org>.

The Association for Computing Machinery (ACM) has just completed a review of its special interest groups (SIGs), and while many SIGs are losing membership and viability, SIGAda has been found to continue to be a viable part of the ACM. ISO and SIGAda will continue to be the key focal points for the distribution of information on Ada 2005 and its predecessors and successors.

There are four *major* Ada compiler vendors: AdaCore (GNAT Pro), Aonix (ObjectAda), Green Hills (AdaMulti), and IBM Rational (Apex). There are also several smaller Ada compiler vendors: DDC-I, Irvine Compiler, OCSystems, RR Software, and SofCheck. Finally, there are several vendors providing tools in support of Ada software development includ-

ing, but not limited to: Grammatech, IPL, LDRA, PolySpace, Praxis, and Vector.

Conclusion

Ada 2005 is with us now, and as users become familiar with and ask for the new features, Ada compiler vendors will respond to the users and implement those features. By no means does the evolution of Ada stop now. The needs of the developers of high-reliability, mission-critical, safety-critical, and high-performance systems will continue to change, and by ISO requirement, as long as Ada is an ISO standard, it will continue to be reviewed and updated periodically (on the order of every five years). Ada is here for the long run, and developers of essential systems for the long run should continue to consider Ada.◆

Special Acknowledgement

I wish to acknowledge and give a special thank you to Dr. David Cook for his early review of this article and commentary. Some of his words appear in the sidebar, and his insight and experience, as always, has been appreciated.

Notes

1. See <www.ansi.org>.
2. See <www.iso.org>.
3. Visit the Sun Developer Network's Java Technology Web site at <http://java.sun.com>.
4. Visit the Microsoft Developer Network's C# Developer Center at <http://msdn.microsoft.com/vcsharp>.
5. Visit the Microsoft Developer's Network Visual Basic Developer Center at <http://msdn.microsoft.com/vbasic>.
6. See ISO/IEC Standard ISO/IEC 8652:1995, "Information Technology – Programming Languages – Ada."
7. See ISO/IEC Standard ISO/IEC 9899:1990, "Programming Languages – C" and ISO/IEC Standard ISO/IEC 9899:1999/Cor 1:2001, "Programming Languages – C – Technical Corrigendum 1."
8. See ISO/IEC Standard ISO/IEC 14882:2003, "Programming Languages – C++."
9. See ISO/IEC Standard ISO/IEC 23270:2003, "Information Technology – C# Language Specification."
10. See ISO/IEC Standard ISO/IEC 23271:2003, "Information Technology – Common Language Infrastructure."
11. Ada 95 is formally identified as ISO/IEC 8652:1995, "Information Technology – Programming Languages – Ada." Minor changes to it were approved and published in June 2001 as ISO/IEC 8652:1995: COR.1: 2001: "Technical Corrigendum to Information Technology – Programming Languages – Ada." Ada 2005 will formally be published as an amendment to ISO/IEC 8652:1995.
12. See Martin Carlisle's Web site at <www.martincarlisle.com> for information on and access to his work on the A# compiler, a set of Ada-oriented utilities, AdaGIDE (an Integrated Development Environment for Ada used at the U.S. Air Force Academy), and Rapid Ada Portable Interface Designer, also covered in this issue of CrossTalk.
13. See ISO/IEC TR 24718:2005, "Information Technology – Programming Languages – Guide for the Use of the Ada Ravenscar Profile in High Integrity Systems."

About the Author

Richard L. Conn is retired, having most recently worked for Microsoft, where he was an Academic Relations Manager (a liaison between the research and product teams at Microsoft and many universities in the United States), and Lockheed Martin, where he was a Software Process Engineer for the C-130J aircraft. Conn has more than 25 years of experience in software development and engineering. He has been involved with Ada since 1979, having been involved in the final stages of the Department of Defense (DoD)-1 language competition (DoD-1 later became Ada). He was a member of the Federal Advisory Board on Ada, and received an award from ACM SIGAda for Outstanding Contributions to the Ada Community. Conn's biography is listed in the 2006 edition of *Marquis' Who's Who in America* and the 2006-2007 edition of *Marquis' Who's Who in American Education*. He has a Master of Science degree in computer science from the University of Illinois Champaign/Urbana, and is an Institute of Electrical and Electronics Engineers Computer Society Certified Software Development Professional.

E-mail: rickconn7@msn.com

Coming Events

September 11-15
PSQT '06 North
Practical Software Quality and Testing
Minneapolis, MN
www.psqtconference.com/2006north

September 11-15
RE06
14th IEEE International Requirements Engineering Conference
Minneapolis/St. Paul, MN
www.re06.org

September 18-21
COMPSAC 2006
30th Annual International Computer Software and Applications Conference
Chicago, IL
http://conferences.computer.org/compsac/2006

September 24-27
ICSM 2006
22nd IEEE International Conference on Software Maintenance
Philadelphia, PA
http://icsm2006.cs.drexel.edu

September 25-27
IDGA
Institute for Defense and Government Advancement
FIREPOWER 2006
Silver Spring, MD
www.idga.org/cgi-bin/templates/genevent.html?topic=228&event=10547&

November 12-16
SIGAda 2006
The Annual International Conference on the Ada Programming Language
Albuquerque, NM
www.sigada.org/conf/sigada2006/

2007
2007 Systems and Software Technology Conference
STC Systems & Software Technology Conference
www.sstc-online.org